

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平11-143688

(43)公開日 平成11年(1999)5月28日

(51)Int.Cl.⁶

G 0 6 F 7/72
G 0 9 C 1/00

識別記号

6 2 0
6 5 0

F I

G 0 6 F 7/72
G 0 9 C 1/00

6 2 0 B
6 2 0 Z
6 5 0 A

審査請求 未請求 請求項の数5 OL (全10頁)

(21)出願番号

特願平9-304758

(22)出願日

平成9年(1997)11月6日

(71)出願人

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72)発明者

岡田 壮一

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72)発明者

島居 直哉

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(74)代理人

弁理士 河野 登夫

最終頁に続く

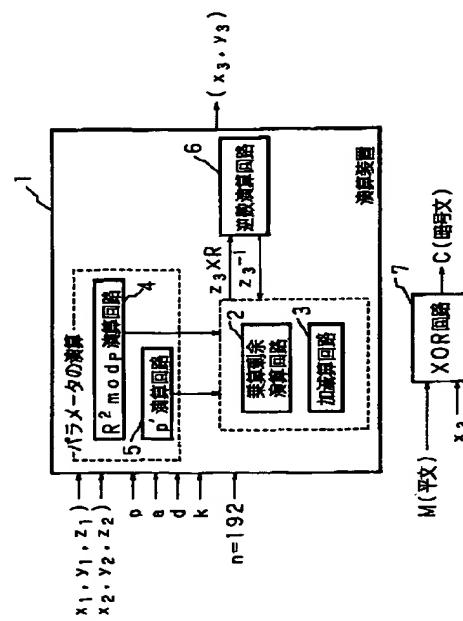
(54)【発明の名称】 演算装置並びにこれを利用したRSA暗号演算装置及び楕円暗号演算装置

(57)【要約】

【課題】 小さい回路規模で演算処理を行える、RSA暗号演算と楕円暗号演算との両方が可能な装置を提供する。

【解決手段】 RSA暗号演算装置に、加減算回路3と逆数演算回路6とを付加することにより、RSA暗号処理と楕円暗号処理との両演算が可能である。乗算剩余演算回路2で、変数を格納するためにレジスタではなくワーキングメモリを使用して、回路規模を小型化を図る。

本発明の原理説明図



BEST AVAILABLE COPY

【特許請求の範囲】

【請求項1】 モンゴメリのアルゴリズムに応じた演算を含む演算処理を行う演算装置において、変数と途中の演算結果とを格納するワーキングメモリを設けたことを特徴とする演算装置。

【請求項2】 モンゴメリのアルゴリズムを利用して、RSA暗号処理を行うRSA暗号演算装置において、変数と途中の演算結果とを格納するワーキングメモリを設けたことを特徴とするRSA暗号演算装置。

【請求項3】 モンゴメリのアルゴリズムを利用して、楕円暗号処理を行う楕円暗号演算装置において、変数と途中の演算結果とを格納するワーキングメモリを設けた乗算剩余演算回路と、モンゴメリ系での演算に必要なパラメータを演算するパラメータ演算回路と、加減算を行う加減算回路と、モンゴメリ系での逆数演算を行う逆数演算回路とを備えることを特徴とする楕円暗号演算装置。

【請求項4】 前記逆数演算回路は、モンゴメリの逆数アルゴリズムにおける2組4個の変数の加算処理を各組毎にそれぞれ並列的に行う2個の加算器を有する請求項3記載の楕円暗号演算装置。

【請求項5】 前記パラメータ演算回路及び／または前記逆数演算回路は、ビットのシフト処理を結線のつなぎ替えにて行うように構成した請求項3記載の楕円暗号演算装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、RSA暗号の演算装置及び楕円暗号の演算装置、並びに、これらの演算装置における各種の演算回路に関する。

【0002】

【従来の技術】 高度情報化社会と呼ばれる現代社会では、コンピュータネットワークを基盤として、ビジネス上の重要な文書・画像情報が電子的な情報という形で伝送通信されて処理される。このような電子情報は、容易に複写が可能である、複写物とオリジナルとの区別が困難であるという性質があり、情報保全の問題が重要視されている。特に、無線を利用したネットワークにおいては、電子情報の傍受が容易なために、これを防止する対策が望まれている。このような対策の有効な手法として、人類の過去の歴史上で主として軍事、外交面に用いられてきた暗号技術が注目されており、種々のネットワークにも導入されつつある。

【0003】 暗号とは、情報の意味が当事者以外には理解できないように情報を変換することである。暗号において、誰でも理解できる元の文（平文）を第三者には意味がわからない文（暗号文）に変換することが暗号化であり、また、暗号文を平文に戻すことが復号化である。この暗号化の過程及び復号化の過程には、それぞれ暗号鍵及び復号鍵と呼ばれる秘密の情報が用いられる。

【0004】 このような鍵を用いる暗号化方式には、秘密鍵暗号系と公開鍵暗号系とが存在する。秘密鍵暗号系は、送信者と受信者とが同じ暗号鍵を持つことによって暗号通信を行う方式であり、送信者はある情報を秘密の暗号鍵に基づいて暗号化して受信者に送り、受信者はこの暗号鍵を用いて暗号を復号して元の情報を得る。これに対して、公開鍵暗号系では、公開されている受信者の公開鍵で送信者が情報を暗号化して送信し、受信者は自分の秘密鍵でその暗号を復号して元の情報を得る。

【0005】 公開鍵暗号系の中で最も有力な暗号系として、1977年にRivest, Shamir及びAdlemanの3人によって発明されたRSA暗号がある。以下、このRSA暗号の原理について簡単に説明する。

【0006】 RSA暗号では、暗号鍵（e, N）、復号鍵（d, N）、平文をM、暗号文をCとした場合、暗号化Eと復号化Dのアルゴリズムは、べき乗剩余演算を用いて次のように表される。

$$C = E(M) = M^e \bmod N$$

$$M = D(C) = C^d \bmod N$$

【0007】 但し、 $e \cdot d \equiv 1 \pmod{\text{LCM}\{(p-1), (q-1)\}}$ 、
 $N = p \cdot q$ (p, q は大きな素数)

e, N: 公開鍵

d: 秘密鍵

LCM (Lowest Common Multiple) : 最小公倍数

【0008】 上記式の場合に、それぞれe回の乗算と除算、d回の乗算と除算を行わなければならず、通常、e, d, M, Nは512ビット以上の大きな整数が用いられ、高速指数演算法を用いても、1回のRSA演算で平均

770回の乗算と剩余演算とを行わなければならない。よって、RSA暗号演算を高速に行うためには、剩余演算の高速化が必要不可欠であり、特に剩余演算は、近似法、モンゴメリのアルゴリズム等、多くの高速化手法が提案されている。

【0009】 RSA暗号に代表される公開鍵暗号系の多くで利用されるべき乗剩余アルゴリズムを高速に処理するためには、1回あたりの剩余アルゴリズムの高速化が要求される。このような観点に基づいて、本発明者等は、剩余演算の高速化を実現するべく、モンゴメリのアルゴリズムを用いた乗算剩余演算装置を提案している

（特開平7-20778号公報）。この乗算剩余演算装置では、モンゴメリ法による剩余演算における繰り返し処理の部分にテーブルを用いて、効率良く剩余演算処理を高速化し、剩余演算の処理速度を向上させる。

【0010】 ここで、モンゴメリのアルゴリズムについて簡単に説明する。モンゴメリのアルゴリズムは、剩余の法N ($N > 1$) と、剩余の法Nと互いに素である基底Rとを用いると、被剩余数Tから $T R^{-1} \bmod N$ の計算が基底Rによる除算のみで行えることを利用して、Nによる除算を用いることなく剩余計算を行うアルゴリズムで

ある。ここで、 N, N', R, R^{-1} 及び T は整数であり、被剰余数 T は $0 \leq T < RN$ 、 R^{-1} は剰余の法 N の上の基底 R の逆数であり、 $R R^{-1} - NN' = 1$ ($0 \leq R^{-1} < N, 0 \leq N' < R$) の関係を満たす。

【0010】更に、この基底 R に2のべき乗数を使用した場合、基底 R による除算をシフト操作に置き換えることができるため、 $T \rightarrow TR^{-1} \bmod N$ の計算の高速処理が可能となる。具体的なアルゴリズムとして、 $T \rightarrow TR^{-1} \bmod N$ のアルゴリズムREDC (T) を以下に示す。但し、このアルゴリズムにおいて、 $(T + mN) / R$ は必ず割り切れる。

$T \rightarrow TR^{-1} \bmod N$ のアルゴリズムREDC (T)

$$\begin{aligned} \text{REDC}(\text{REDC}(T) \times (R^2 \bmod N)) \\ = (TR^{-1} \bmod N) (R^2 \bmod N) R^{-1} \bmod N \\ = TR^{-1} \times R^2 \times R^{-1} \bmod N \\ = T \bmod N \end{aligned}$$

10

このようにして、簡単に $T \bmod N$ を求めることができる。

【0012】

【発明が解決しようとする課題】特開平7-20778号公報に示した乗算剰余演算装置では、モンゴメリのアルゴリズムでの変数はレジスタに格納しているが、RSA演算にあっては、演算処理の高速化に加えてその演算装置の回路規模の小型化も重要な問題である。

【0013】ところで、RSA暗号は、公開鍵暗号の中でも、有力で実用的な暗号形式であると言われており、RSA暗号用の演算チップの開発が積極的に進められている。しかし、近年、RSA暗号の強度の低下について言及されるようになり、RSA暗号に代わる公開鍵暗号として、楕円暗号が注目されている。

【0014】楕円暗号の基本演算について、以下に記述する。有限体 F_p (p は $p > 3$ の素数)において、 F_p 上の楕円曲線 $E(F_p)$ は、

$$y^2 = x^3 + ax + b$$

但し、 a, b は $0 \leq a, b < p$ なる整数であり、

$$4a^3 + 27b^2 \neq 0 \pmod{p}$$

で表され、楕円曲線上の点とは、 $E(F_p)$ 上の点の組と無限遠点 O とのことを言う。

【0015】この楕円曲線 $E(F_p)$ 上の点は、以下の加算ルールで加算を行う。

$$(1) O + O = O$$

$$(2) (x, y) + O = (x, y)$$

$$(3) (x, y) + (x, -y) = O$$

(点 (x, y) と点 $(x, -y)$ とは逆数の関係にある)

(4) 逆数の関係にない2点 $(x_1, y_1), (x_2, y_2)$ に対して、加算結果を (x_3, y_3) とする。

① $(x_1, y_1) \neq (x_2, y_2)$ の場合

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

但し、 $x_3 = \lambda^2 - x_1 - x_2$

$$*m = (T \bmod R) N' \bmod R$$

$$t = (T + mN) / R$$

もし、 $t < N$ の場合には t になり、それ以外は $t - N$ 即ち、

$$\text{REDC}(T) = t \quad (t < N)$$

$$\text{REDC}(T) = t - N \quad (t \geq N)$$

【0011】1回のREDCでは、剰余 $T \bmod N$ ではなく $TR^{-1} \bmod N$ が求められるだけである。そこで、剰余 $T \bmod N$ を求めるためには、以下に示すように、REDC (T) と予め求めておいた $R^2 \bmod N$ との積で、再びREDCを行えばよい。

10

$$y_3 = \lambda (x_1 - x_3) - y_1$$

$$\lambda = (y_1 - y_2) / (x_1 - x_2)$$

② $(x_1, y_1) = (x_2, y_2)$ の場合

2 $(x_1, y_1) = (x_3, y_3)$

但し、 $x_3 = \lambda^2 - 2x_1$

$$y_3 = \lambda (x_1 - x_3) - y_1$$

$$\lambda = (3x_1^2 + a) / 2y_1$$

【0016】次に、楕円曲線上の点を高速に加算する手法を説明する。まず、楕円曲線上の点を三次元空間で表現する。即ち、 (x, y) を $(x/z, y/z)$ と対応させる(三次元投影法)。三次元で表した点の座標をモンゴメリ系に変換し、加算ルールに従って点の加算(各座標では、 $\bmod p$ 上で加減乗算)を行う。最後に、 z 座標の逆数 $z^{-1} \bmod p$ を求め、それを x 座標及び y 座標と乗算して、点の加算結果を得る。

30

【0017】上述したような楕円暗号演算を行う場合上記の加算ルール(4)のように、毎回、除算を行わなければならないし、三次元投影法にモンゴメリ法を適用して加算処理を行う手法においても、最後に、逆数を計算しなければならない。RSA暗号演算を行う回路構成では、除算、逆数演算を行うことができず、ソフトウェアでこれらの演算処理を行うと、長時間要するという問題がある。例えば192ビットの逆数演算を10000回行う場合、拡張ユーリッドでは約12秒、モンゴメリイバースアルゴリズムでは、約7.3秒かかる(Pentium 133MHz)。

40

【0018】本発明は斯かる事情に鑑みてなされたものであり、小さい回路規模で演算処理が可能であり、変数のビット長の拡張にも対応できる演算装置、及び、RSA暗号演算装置を提供することを目的とする。

50

【0019】本発明の他の目的は、RSA暗号演算装置に、加減算回路と逆数演算回路とを付加することにより、楕円暗号演算を可能とする楕円暗号演算装置を提供することにある。

【0020】本発明の更に他の目的は、小さい回路規模で演算時間が早い楕円暗号演算装置を提供することにある。

【0021】

【課題を解決するための手段】請求項1に係る演算装置は、モンゴメリのアルゴリズムに応じた演算を含む演算処理を行う演算装置において、変数と途中の演算結果とを格納するワーキングメモリを設けたことを特徴とする。

【0022】請求項2に係るRSA暗号演算装置は、モンゴメリのアルゴリズムを利用して、RSA暗号処理を行うRSA暗号演算装置において、変数と途中の演算結果とを格納するワーキングメモリを設けたことを特徴とする。

【0023】請求項3に係る楕円暗号演算装置は、モンゴメリのアルゴリズムを利用して、楕円暗号処理を行う楕円暗号演算装置において、変数と途中の演算結果とを格納するワーキングメモリを設けたことを特徴とする。

【0024】請求項4に係る楕円暗号演算装置は、請求項3において、前記逆数演算回路は、モンゴメリの逆数アルゴリズムにおける2組4個の変数の加算処理を各組毎にそれぞれ並列的に行う2個の加算器を有することを特徴とする。

【0025】請求項5に係る楕円暗号演算装置は、請求項3において、前記パラメータ演算回路及び／または前記逆数演算回路は、ビットのシフト処理を結線のつなぎ替えにて行うように構成したことを特徴とする。

【0026】図1は、本発明の原理説明図である。図において1は、RSA暗号及び楕円暗号の演算を行う演算装置である。演算装置1は、モンゴメリ系での乗算を行う乗算剩余演算回路2と、加算・減算を行う加減算回路3と、モンゴメリ系での演算に必要なパラメータ $R^2 \bmod p$ を計算する $R^2 \bmod p$ 演算回路4と、モンゴメリ系での演算に必要なパラメータ p' を計算する p' 演算回路5と、入力変数 z の逆数 z^{-1} を求める逆数演算回路6

*
乗算 $a \times b \rightarrow \text{REDC}(a \times b)$

加算 $a + b \rightarrow a + b > p$ の場合 $a + b - p$
それ以外 $a + b$

減算 $a - b \rightarrow a > b$ の場合 $a - b$
それ以外 $a - b + p$

$\ast((x_3 \times R \bmod p) \times Z_3^{-1})$ と $\text{REDC}((y_3 \times R \bmod p) \times Z_3^{-1})$ とを計算する。逆数演算回路6は、モンゴメリ系での逆数演算アルゴリズム（モンゴメリインバースアルゴリズム）に基づいて演算を行う。演算結果は、

$$\text{REDC}((x_3 \times R \bmod p) \times Z_3^{-1}) = x_3 \times R \times Z_3^{-1} \times R^{-1} \bmod p$$

*とを有する。また、7は排他的論理和を求めるXOR回路である。

【0027】図1の構成は、RSA暗号演算装置に加減算回路3及び逆数演算回路6を付加したものであって、RSA暗号演算を行えるだけでなく楕円暗号演算も行える構成である。

【0028】次に、本発明の動作原理について説明する。モンゴメリ系への変換処理は、乗算剩余演算回路2と $R^2 \bmod p$ 演算回路4とを用いて行う。与えられる開

始点 $P (x_0/z_0, y_0/z_0)$ とすると、

$$X_0 = \text{REDC}(x_0 \times R^2 \bmod p)$$

$$= x_0 \times R^2 \times R^{-1} \bmod p$$

$$= x_0 \times R \bmod p$$

$$Y_0 = \text{REDC}(y_0 \times R^2 \bmod p)$$

$$= y_0 \times R^2 \times R^{-1} \bmod p$$

$$= y_0 \times R \bmod p$$

$$Z_0 = \text{REDC}(z_0 \times R^2 \bmod p)$$

$$= z_0 \times R^2 \times R^{-1} \bmod p$$

$$= z_0 \times R \bmod p$$

となる。但し、この変換は、開始点だけについて行う。

【0029】モンゴメリ系への変換後は、加算ルールに従って加算を行う。その結果、以下の式に基づく演算を $\bmod p$ の上で行うことになる。

【0030】点 $(x_1, y_1, z_1), (x_2, y_2, z_2)$ の演算結果を点 (x_3, y_3, z_3) とすると、
(1) 加算（異なる2点）

$$x_3 = A B$$

$$y_3 = C (A^2 x_1 z_2 - B) - A^3 y_1 z_2$$

$$z_3 = A^3 z_1 z_2$$

30 但し、 $A = x_2 z_1 - x_1 z_2, B = C^2 z_1 z_2 - A^2 D$

$$C = y_2 z_1 - y_1 z_2, D = x_2 z_1 + x_1 z_2$$

(2) 2倍算（同じ点）

$$x_3 = 2 A B$$

$$y_3 = C (4 D - B) - 8 E^2$$

$$z_3 = 8 A^3$$

但し、 $A = y_1 z_1, B = C^2 - 8 D, C = 3 x_1 + a z_1^2, D = x_1 E, E = y_1 A$

で表される。実際には、変数 a, b に対して、

の演算を、乗算剩余演算回路2と加減算回路3とを用いて行う。

【0031】最後に、逆数演算回路6に $z_3 \times R$ を入力して、 $z_3 \times R \bmod p$ のモンゴメリインバース $Z_3^{-1} = z_3^{-1} \times R^{-1} \times R \bmod p = z_3^{-1} \bmod p$ を求めて、 $\text{REDC} \ast$

$$\text{REDC}((y_3 \times R \bmod p) \times Z_3^{-1})$$

$$= x_3 \times z_3^{-1} \bmod p$$

$$= y_3 \times R \times z_3^{-1} \times R^{-1} \bmod p$$

$$= y_3 \times z_3^{-1} \bmod p$$

リを利用する。

となる。

【0032】以上の演算処理により、橜円曲線上の開始点Pに対して、kPを得る。そして実際の暗号化・復号化処理は、次のような手順で行う。図2は、送信者Aと受信者Bにおける暗号化・復号化処理の関係を示す模式図である。

【0033】(送信者A側での暗号化) 乱数kに対して、

$$kP = (x_1, y_1)$$

$$kQ_B = (x_2, y_2)$$

を計算する。平文Mとx2との排他的論理和をXOR回路7で計算し、暗号文Cを得る。(x1, y1, C)を受信者Bへ送る。

【0034】(送信者B側での復号化) $d_B(x_1, y_1) = d_B kP = k d_B P = k Q_B = (x_2, y_2)$ を得る。暗号文Cとx2との排他的論理和をXOR回路7計算して、平文Mを得る。

【0035】上述したような構成において、本発明の乗算剩余演算回路2は、従来の変数格納用のレジスタの代わりにワーキングメモリ(変数格納用とワーク用)を用いている。よって、従来より回路規模を小さくでき、また、変数のビット長の拡張にも対応できる。

【0036】また、本発明の逆数演算回路6は、2つの加算器を有し、2組の変数(1組あたり2つの変数で合計4つの変数)に対して、1サイクルで同時に演算を行う。よって、演算時間の高速化を図れる。

【0037】また、本発明のp'演算回路5と逆数演算回路6において、シフト演算処理を結線のつなぎ替えにて行うようにする。よって、これらの回路の規模を小さくできる。

【0038】

【発明の実施の形態】以下、本発明をその実施の形態を示す図面を参照して具体的に説明する。図3～図7は、前述した図1における乗算剩余演算回路2、加減算回路3、 $R^2 \bmod p$ 演算回路4、 p' 演算回路5及び逆数演算回路6の各回路のハードウェア構成を示す図である。図3～図7を参照して、各回路について具体的に説明する。

【0039】(乗算剩余演算回路) 図3は、乗算剩余演算回路2の内部構成の一例を示す。図においてFFはフリップフロップを示し、64×64乗算器は64ビット×64ビットの乗算器、64加算器は64ビットの加算器をそれぞれ示す。本発明では、レジスタの代わりにワーキングメモ

$$= x_3 \times z_3^{-1} \bmod p$$

$$= y_3 \times R \times z_3^{-1} \times R^{-1} \bmod p$$

$$= y_3 \times z_3^{-1} \bmod p$$

リを利用する。

【0040】本例の乗算剩余演算回路2では、途中の計算結果を格納するワーキングメモリの容量を小さくするために、乗数を分割して乗算剩余演算を行う方式を採用する。例えば、被乗数A(192ビット)、乗数B(192ビット)とした場合、乗数Bを64ビット単位に分割して部分積毎に剩余を求めていく。

【0041】モンゴメリ系では、下位64ビットがすべて0になるようにpの倍数を加算するために、通常の部分積剩余演算とは異なり、下位64ビットから計算していく必要がある。即ち、 $A \times B_0$ の乗算を行い、その結果の値(192+64ビット)の下位64ビットが0となるように、pの倍数を足していく。pの倍数は、下位64ビットの値に p' をかけた結果の下位64ビットである。次に、 $A \times B_1, A \times B_2$ に同様の計算を行う。

【0042】以下に、アルゴリズムを示す。

$$i = 0; i < 3; ++$$

$$i \leftarrow 0, W \leftarrow 0$$

$$W \leftarrow A \times B_i + W$$

$$W \leftarrow W + (((W \bmod 2^{64}) \times p') \bmod 2^{64}) \times p$$

$$W \leftarrow W / 2^{64}$$

【0043】(加減算回路) 図4は、加減算回路3の内部構成の一例を示す。図において、FFはフリップフロップ、64FFは64ビットのフリップフロップ、64加算器は64ビットの加算器をそれぞれ示す。

【0044】加減算回路3において、コマンドレジスタに加算か減算かを設定する。設定した値(0または1)によって第1、第2のセレクタの制御が、下記表1のように行われる。

【0045】

【表1】

表 1

	第1のセレクタ	第2のセレクタ
加算の場合	b	-p
減算の場合	-b	p

【0046】a, b, pを64ビット単位で下位から演算を行い、各回の演算の結果を、下記表2のように第4、第5のワーキングメモリに格納していく。このような演算・格納処理を3回行う。

【0047】

【表2】

	第4のワーキングメモリ	第5のワーキングメモリ
加算の場合	$a + b$	$a + b - p$
減算の場合	$a - b$	$a - b + p$

【0048】3回目の演算後、第1、第2の64加算器でのキャリー情報とコマンドレジスタでの設定情報により、第4のワーキングメモリと第5のワーキングメモリとの何れの結果を選択するかを、下記表3のように、判*10

*定器にて判定する。

【0049】

【表3】

表 3

	キャリー1	キャリー2	判 定
加算の場合	1	X	第5のワーキングメモリ
	0	1	第5のワーキングメモリ
	0	0	第4のワーキングメモリ
減算の場合	1	X	第4のワーキングメモリ
	0	X	第5のワーキングメモリ

【0050】($R^2 \bmod p$ 演算回路) 図5は、 $R^2 \bmod p$ 演算回路4の内部構成の一例を示す。図において、F Fはフリップフロップ、64加算器は64ビットの加算器をそれぞれ示す。

ステップ1 $Y \leftarrow R \bmod p, T \leftarrow 192$

ステップ2 $Y \leftarrow Y \times 2 \bmod p$

ステップ3 $T \leftarrow T - 1$, もし、 $T = 0$ の場合はステップ4へ
それ以外はステップ2へ

ステップ4 $R^2 \bmod p \leftarrow Y$

なる計算を行うことが一般的である。ここでは、計算時 30★ことにする。つまり、間の短縮化を図るために、2ビット単位での処理を行う★

ステップ1 $Y \leftarrow R \bmod p, T \leftarrow 192$

ステップ2 $Y \leftarrow Y \times 4 \bmod p$

ステップ3 $T \leftarrow T - 2$, もし、 $T = 0$ の場合はステップ4へ
それ以外はステップ2へ

ステップ4 $R^2 \bmod p \leftarrow Y$

なる計算を行う。

【0052】このような $Y \times 4 \bmod p$ の計算をハードウェアで行う場合、なにも引かない、 $-p$ 、 $-2p$ 、 $-3p$ の剩余演算が必要となり、並列に行うには、4個の加算器が必要になる。本例の $R^2 \bmod p$ 演算回路4では、このような演算を表4に示すようなアルゴリズムを用い

*【0051】ここで、 p の値を192ビットとする。即ち、 $R = 2^{192}$ となる値を用い、 $R^2 \bmod p$ 演算回路4では、 $2^{384} \bmod p$ を計算することになる。基本的な演算方法としては、

* 【0053】

【表4】

て2個の加算器（第1、第2の64加算器）で行い、回路規模の削減と計算の高速化とを実現している。つまり、被除数Yの上位3ビット、除数Pの上位2ビットに応じて、以下の表4のような処理を行う。

【0053】

【表4】

BEST AVAILABLE COPY

Y	P	処理	結果
0 0 0	1 X	剩余計算を行わずに2ビットシフト	± 0
0 0 1	1 X	2ビットシフト後-p	-p
0 1 0	1 0	1ビットシフト後-p または 2ビットシフト後-p	-pまたは-2p
0 1 0	1 1	2ビットシフト後-p	-p
0 1 1	1 0	1ビットシフト後-p さらに 1ビットシフト後-p	-2p または-3p
0 1 1	1 1	1ビットシフト後-p または 2ビットシフト後-p	-pまたは-2p
1 XX	1 X	1ビットシフト後-p さらに 1ビットシフト後-p	-2p または-3p

【0054】なお、1024ビットのRSA演算の場合には、 $R = 2^{1024}$ として $R^2 \bmod p$ 演算回路4での演算を行う。

【0055】(p' 演算回路) 図6は、 p' 演算回路5の内部構成の一例を示す。本発明では、1ビットのシフト演算処理が結線のつなぎ替えにて行われるように構成されており、回路規模の小型化を図っている。

【0056】モンゴメリ系では、 $R \times R^{-1} - p \times p' = 1$ となる p' を用いて演算を行う。 p' 演算回路5でこ*

ステップ1 $Y \leftarrow 0, T \leftarrow 0, p' \leftarrow 0$
 ステップ2 $Y \bmod 2 = 0$ の場合は $Y \leftarrow Y + p, p' \leftarrow p' + 2T$
 ステップ3 $Y \leftarrow Y / 2$
 ステップ4 $T = 63$ の場合は終了 それ以外はステップ2へ

【0058】この演算は、下位のビットから p のべき数倍を加えていくことにより、下位64ビットがすべて1になるような p を求めていく演算である。

【0059】(逆数演算回路(モンゴメリインバース演算回路)) 図7は、逆数演算回路6の内部構成の一例を示す。図において、FFはフリップフロップ、64FFは64ビットのフリップフロップ、64加算器は64ビットの加算器をそれぞれ示す。本発明では、第1の64加算器、第2の64加算器の2個の加算器を有し、後述するような u, v の2変数に対する第1の64加算器での処理と、後述するような r, s の2変数に対する第2の64加算器での処理とを、1サイクルで同時に進行ようにして、演算時間の短縮化を図っている。

【0060】橢円曲線上の点の加算処理の結果、得られる座標値は、

X座標 $x_3 \times R \bmod p$

Y座標 $y_3 \times R \bmod p$

Z座標 $z_3 \times R \bmod p$

である。ここで、Z座標の逆数 z_3^{-1} を求めて、最終的に座標 $(x_3 \times z_3^{-1}, y_3 \times z_3^{-1})$ を求めるのであるが、逆数演算に以下のモンゴメリインバースアルゴリズムを用いて z_3^{-1} を求める。

【0061】[アルゴリズム]

入力: a, b, n (a は奇数 ($a > b > 0, n$ は a のビット数))

*の p' を求める。上記式の両辺に $\bmod R$ をとることにより、以下のようにになる。

$$p \times p' \equiv R - 1 \pmod{R}$$

【0057】 R は2のべき乗の値をとるため、右辺は右倍しても R 以下のビット(本暗号ハードウェア下位64ビット)に関しては、 $p \times p'$ の値がすべて1になるように、 p' を選べば良い。 p' を求めるアルゴリズムは以下のとおりである。

出力: 「互いに素でない」または $b^{-1} \times 2^n \bmod a$
 第1フェーズ

$u \leftarrow a, v \leftarrow b, r \leftarrow 0, s \leftarrow 1$
 $k \leftarrow 0$

30 $v = 0$ になるまで、以下を繰り返す。

u が偶数の場合 $u \leftarrow u / 2, s \leftarrow 2s$

u が奇数の場合

v が偶数のとき

$v \leftarrow v / 2, r \leftarrow 2r$

v が奇数のとき

$u > v$ の場合 $u \leftarrow (u - v) / 2, r \leftarrow r + s, s \leftarrow 2s$

$u \leq v$ の場合 $v \leftarrow (v - u) / 2, s \leftarrow r + s, r \leftarrow 2r$

40 $k \leftarrow k + 1$

$u \neq 1$ なら 「互いに素でない」を返す。

$r \geq a$ なら $r \leftarrow r - a$

第2フェーズ

$i = 1; i \leq k - n; ++$

r が偶数の場合 $r \leftarrow r / 2$

r が奇数の場合 $r \leftarrow (r + a) / 2$

$a - r$ を返す。

【0062】 $R = 2^{192}$ より $n = 192$ (固定)となり、 $k < n$ の場合が出てくる。そのため、上記アルゴリズムの第2フェーズを以下のように、拡張する。

BEST AVAILABLE COPY

第2フェーズ

k ≥ n の場合

i = 1 ; i ≤ k - n ; ++

r が偶数のとき r ← r / 2

r が奇数のとき r ← (r + a) / 2

k < n の場合

i = 1 ; i ≤ k - n ; ++

r ← 2 r

r > a のとき r ← r - a

それ以外は そのまま

a - r を返す。

【0063】また、逆数を計算するアルゴリズムとして、拡張ユークリッドが知られているが、モンゴメリインバースアルゴリズムでは、計算の途中に負の値をとらないため、ハードウェアでの実現に有効である。

【0064】上述した本例の乗算剩余演算回路2、加減算回路3、 $R^2 \bmod p$ 演算回路4及び p' 演算回路5の回路規模の合計は約5万8千ゲートであり、これに上述の逆数演算回路6の約5千ゲートの回路規模の追加により、楕円暗号演算が可能となる。また、上述した本回路構成での楕円暗号演算の処理時間は、約60msec (クロック 5MHz, データ長192ビット) であり、逆数演算回路 (モンゴメリインバース演算回路) 6の処理時間は、約0.6 msecと評価できる。

【0065】

* 【発明の効果】以上説明したように、本発明によれば、乗算剩余演算回路に、加減算回路と逆数演算回路とを附加することにより、RSA暗号演算と楕円暗号演算との両演算が可能となる。また、レジスタの代わりにワーキングメモリを使って回路規模の小型化を実現することにより、演算チップの省スペース化という効果がある。

【図面の簡単な説明】

【図1】本発明の原理説明図である。

【図2】暗号化・復号化処理の関係を示す模式図である。

【図3】乗算剩余演算回路の構成図である。

【図4】加減算回路の構成図である。

【図5】 $R^2 \bmod p$ 演算回路の構成図である。【図6】 p' 演算回路の構成図である。

【図7】逆数演算回路 (モンゴメリインバース演算回路) の構成図である。

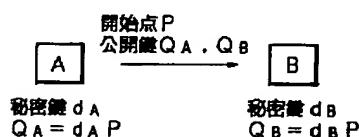
【符号の説明】

- 1 演算装置
- 2 乗算剩余演算回路
- 3 加減算回路
- 4 $R^2 \bmod p$ 演算回路
- 5 p' 演算回路
- 6 逆数演算回路 (モンゴメリインバース演算回路)
- 7 XOR (Exclusive OR) 回路

*

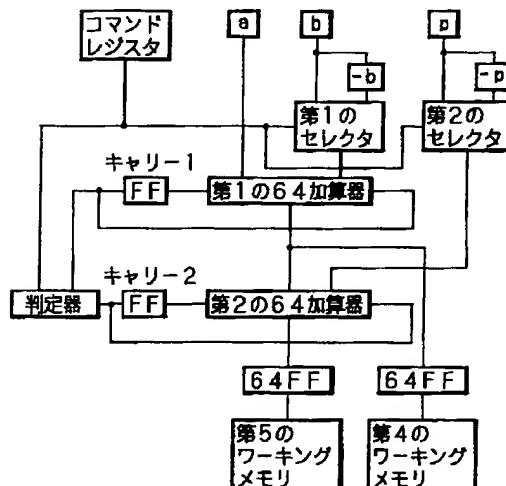
【図2】

暗号化・復号化処理の関係を示す模式図



【図4】

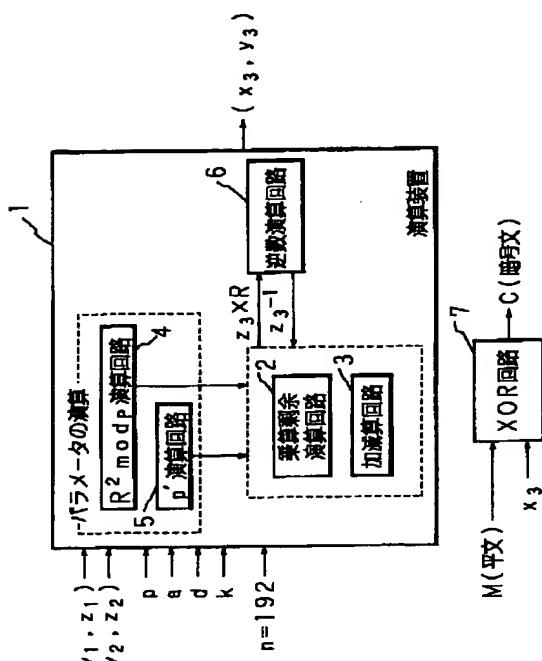
加減算回路の構成図



【四 1】

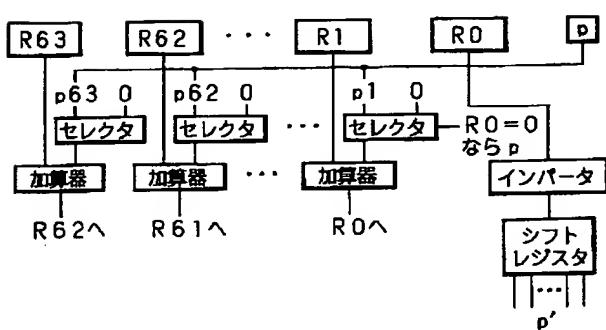
[図3]

本発明の原理説明図

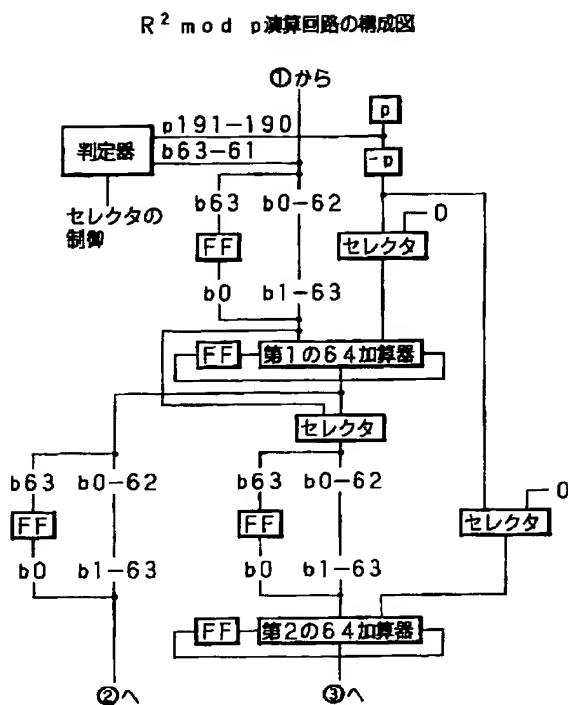


【図6】

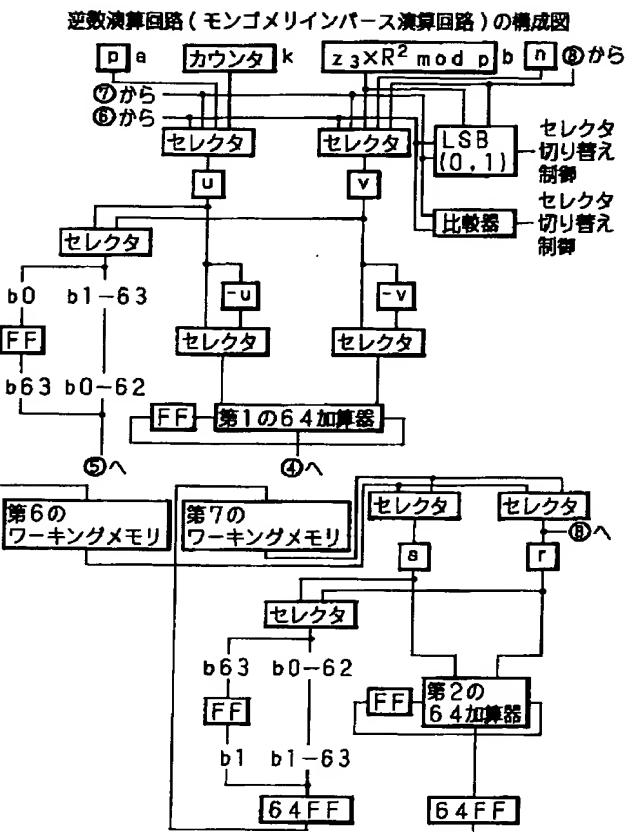
p' 演算回路の構成図



【図5】



[図 7]



フロントページの続き

(72) 発明者 長谷部 高行
神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72) 発明者 武仲 正彦
神奈川県川崎市中原区上小
1号 富士通株式会社内